User Interaction Optimization for an Evolving Classifier of Handwritten Gesture Commands

Manuel Bouillon, Eric Anquetil, Peiyu Li, Grégoire Richard

User Interaction Optimization for an Evolving Classifier of Handwritten Gesture Commands

Manuel BOUILLON, Eric ANQUETIL, PeiYu LI, Grégoire RICHARD
Université Européenne de Bretagne, France
INSA de Rennes, Avenue des Buttes de Coesmes, F-35043 Rennes
IRISA, CNRS UMR 6074, Campus de Beaulieu, F-35042 Rennes
{manuel.bouillon, eric.anquetil, pei-yu.li, gregoire.richard}@irisa.fr

***Abstract*—Touch sensitive interfaces enable new interaction methods, like using gesture commands. The use of gesture commands give rise to a cross-learning situation where the user has to learn and memorize the command gestures and the classifier has to learn and recognize drawn gestures. To easily memorize more than a dozen of gesture commands, it is important to be able to customize them. The classification task associated with the use of customized gesture commands is complex because the classifier only has very few samples per class to start learning from. We thus need an evolving recognition system that can start from very few data samples and that will learn incrementally to achieve good performance after some using time. This paper presents the impact of using rejection based user interactions to supervise the on-line training of the evolving classifier. The objective is to obtain a gesture command system that cooperates as best as possible with the user, to learn from its mistakes without soliciting him too often. To detect confusing classes we apply confusion reject principles to our evolving recognizer, which is based on a first order fuzzy inference system. A significant user experiment has been performed on 63 persons that validates our approach. This user experiment shows the interest of optimizing user interactions by taking into account the confusion detection capability of our recognition system.**

I. INTRODUCTION

With the increasing use of touch sensitive screens, human-computer interactions are evolving. New interaction methods have been invented to take advantage of the new potential of interaction that those new interfaces offer. Among them, a new concept has recently appeared: to associate commands to gestures. Gesture commands [1][2] enable users to execute various actions simply by drawing symbols. Previous studies [3] have shown that enabling customization is essential to help users' memorization of gestures. To use such gesture commands, a handwritten gesture recognition system is required. Moreover, if gestures are personalized, the classifier has to be flexible and able to learn with few data samples.

As we can't expect users to draw much more than a few gesture samples per class, the recognition engine must be able to learn with very few data samples. Some template matching classifiers exist, like the $1 classifier [4] for instance, that don't require much training. However, such simple systems don't evolve with the user writing style. For example, novice users usually draw gestures slowly and carefully, but as they become more and more expert, users draw their gestures more fluidly and rapidly. In that case, we want the classifier to adapt to the user, and not the other way round. More flexibility in a recognizer requires an on-line system, *ie* a system that learn on the run-time data flow.

Evolving classification systems have appeared in the last decade to meet the need for recognizers that work in changing environments. They use incremental learning to adapt to the data flow and cope with class adding (or removal) at run-time. This work uses such an evolving recognizer, *Evolve* [5], which is a first order fuzzy inference system. It can start learning from few data and then learns incrementally in real time from the data flow that it tries to recognize, to adapt its model and to improve its performance during its use.

The on-line learning algorithm is a supervised algorithm that requires labeled data. In the context of gesture command recognition, the only way of knowing the true label of a gesture is to interact with the user. However, soliciting the user after each command cancel the very interest of gesture commands! Our method consists in offering a correction mechanism to the user, and to implicitly validate the recognition in absence of correction. However, if the user fails to correct a recognition error, the classifier will learn with a wrong label and deteriorate its model. In this paper, we study the use of confusion reject to explicitly solicit the user, to correct or validate the recognized label, when the confidence of the classifier is low. In this way, the system learning focuses on complex data sample, from which it is very beneficial to learn from, with the correct label. Moreover, rejecting data samples with low confidence allow to avoid errors by asking confirmation to the user, and save him from canceling/undoing a wrong command before re-doing the intended one. Our objective is to handle as best as possible the cooperation between the user and the command gesture system to optimize the command gesture use and the classifier training, but without soliciting the user too often.

A sensitive issue is to avoid the "out-of-the-loop performance problem" [6]. This problem is the consequence of system automation where the operator loses direct control. This situation can have harmful consequences like vigilance decrements or complacency. To avoid this problem, [7] propose to provide feedback to the operator on the automated task and the possibility to take control in case of failure. In this work, automatic system is the evolving recognition engine. Rejection of complex data that are hard to recognize by the classifier allow to inform the user of system difficulties and to explicitly ask him to take control and correct the system. We expected these rejection based interactions to improve user supervision of the recognizer. We will show in the experiments that it's not so easy for all interactive aspects. In fact, if we consider spontaneous corrections from users (not invoked by

the system), they decrease because of users over confidence in system capability.

If there is a lot of work in the literature on pattern recognition and handwritten gesture classification, very few approach are evaluated in a real applicative context. We conducted a complete experiment to evaluate our approach in a real context of gesture commands utilization, so that we obtain realistic results from users as well as from the recognizer. We have considered two groups of about thirty users, and both groups had to define and use gesture commands in a testing application. A reference group with spontaneous corrections only, and a second with spontaneous corrections and rejection based user interactions, to study its impact on the on-line training of the classifier.

This paper is organized as follows. The Section II presents the architecture of our evolving classifier. We explain in section III the principle of the conflict detection mechanism. Then, we present a realistic user experimentation in Section IV, to analyze the impact of different kinds of user interaction methods and to demonstrate the benefits of the presented approach. Section V concludes and discusses future work.

II. System Architecture

In this work we use a Fuzzy Inference Systems (FIS) [8], with first order conclusion structure [9]. FIS have demonstrated their good performances for incremental classification of changing data flows [10]. Moreover, they can easily be trained on-line – in real time – and have a good behavior with new classes. In this section, we present the architecture of the evolving FIS *Evolve* [5] designed during our past work and that we that we use to recognize our gesture commands.

Fuzzy Inference Systems consist of a set of fuzzy inference rules like the following rule example.

$$\textbf{Rule}^{(i)} : \textbf{IF } \mathbf{x} \text{ is close to } C^{(i)} \tag{1}$$

$$\textbf{THEN } \hat{\mathbf{y}}^{(i)} = (\hat{\mathbf{y}}_1^{(i)}; \ldots; \hat{\mathbf{y}}_c^{(i)})^\top \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the feature vector, $C^{(i)}$ the fuzzy prototype associated to the *i*-th rule and $\hat{\mathbf{y}}^{(i)\top} \in \mathbb{R}^c$ the output vector. Rule premises are the fuzzy membership to rule prototypes, which are clusters in the input space. Rule conclusions are fuzzy membership to all classes, that are combined to produce the system output.

A. Premise Structure

Our model uses rotated hyper-elliptical prototypes that are each defined by a center $\boldsymbol{\mu}^{(i)} \in \mathbb{R}^n$ and a covariance matrix $\Sigma^{(i)} \in \mathbb{R}^{n \times n}$ (where n is the number of features).

The activation degree $\alpha^{(i)}(\mathbf{x})$ of each fuzzy prototype is computed using the multivariate normal distribution.

B. Conclusion Structure

In a first order FIS, rule conclusions are linear functions of the input:

$$\hat{\mathbf{y}}^{(i)\top} = (l_1^{(i)}(\mathbf{x}); \ldots; l_c^{(i)}(\mathbf{x})) \tag{3}$$

$$l_k^{(i)}(\mathbf{x}) = \mathbf{x}^\top \cdot \boldsymbol{\theta}_k^{(i)} = \theta_{0,k}^{(i)} + \theta_{1,k}^{(i)} \cdot x_1 + \cdots + \theta_{n,k}^{(i)} \cdot x_n \tag{4}$$

The *i*-th rule conclusion can be reformulated as:

$$\hat{\mathbf{y}}^{(i)\top} = \mathbf{x}^\top \cdot \Theta^{(i)} \tag{5}$$

with $\Theta^{(i)} \in \mathbb{R}^{n \times c}$ the matrix of the linear functions coefficients of the *i*-th rule:

$$\Theta^{(i)} = (\boldsymbol{\theta}_1^{(i)}; \ldots; \boldsymbol{\theta}_c^{(i)}) \tag{6}$$

C. Inference Process

The inference process consists of three steps:

1) Activation degree is computed for every rule and then normalized as follows:
$$\alpha^{(i)}(\mathbf{x}) = \frac{\alpha^{(i)}(\mathbf{x})}{\sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x})} \tag{7}$$
where r is the number of rules.
2) Rules outputs are computed using Equation 5 and system output is obtained by sum-product inference:
$$\hat{\mathbf{y}} = \sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x}) \cdot \hat{\mathbf{y}}^{(k)} \tag{8}$$
3) Predicted class is the one corresponding to the highest output:
$$\text{class}(\mathbf{x}) = \arg \max_{k=1}^{c} (\hat{y}_k) \tag{9}$$

D. Incremental Learning Process

Let $\mathbf{x}_i$ $(i = 1..t)$ be the *i*-th data sample, M_i the model at time i, and f the learning algorithm. The incremental learning process can be defined as follows:

$$M_i = f(M_{i-1}, \mathbf{x}_i) \tag{10}$$

whereas a batch learning process would be:

$$M_i = f(\mathbf{x}_1, \ldots, \mathbf{x}_i) \tag{11}$$

In our recognizer *Evolve* [5], both rule premises and conclusions are incrementally adapted:

1) Rule prototypes are statistically updated to model the run-time data:
$$\boldsymbol{\mu}_t^{(i)} = \frac{(t-1) \cdot \boldsymbol{\mu}_{t-1}^{(i)} + \mathbf{x}_t}{t} \tag{12}$$
$$\Sigma_t^{(i)} = \frac{(t-1) \cdot \Sigma_{t-1}^{(i)} + (\mathbf{x}_t - \boldsymbol{\mu}_{t-1}^{(i)})(\mathbf{x}_t - \boldsymbol{\mu}_t^{(i)})}{t} \tag{13}$$
2) Rule conclusions parameters are optimized on the data flow, using the Recursive Least Squares (RLS) algorithm:
$$\Theta_t^{(i)} = \Theta_{t-1}^{(i)} + \alpha^{(i)} \mathrm{C}_t^{(i)} \mathbf{x}_t (\mathbf{y}_t^\top - \mathbf{x}_t^T \Theta_{t-1}^{(i)}) \tag{14}$$
$$\mathrm{C}_t^{(i)} = \mathrm{C}_{t-1}^{(i)} - \frac{\mathrm{C}_{t-1}^{(i)} \mathbf{x}_t \mathbf{x}_t^\top \mathrm{C}_{t-1}^{(i)}}{\frac{1}{\alpha^{(i)}} + \mathbf{x}_t^\top \mathrm{C}_{t-1}^{(i)} \mathbf{x}_t} \tag{15}$$

New rules, with their associated prototypes and conclusions, are created by the incremental clustering method *eClustering* [11] when needed.

Figure 1 represents a FIS with first order conclusion structure as a radial basis function (RBF) neural network.

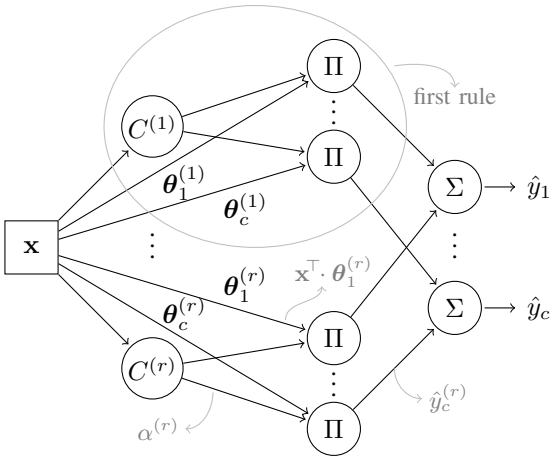


Fig. 1. First order FIS as a radial basis function (RBF) neural network

III. Confusion Based User Interaction

In the context of gesture commands, users initialize the system with a few gestures per class (three in our experimentation). To improve gesture command recognition, the classifier learns incrementally during its use. The on-line learning algorithm used to train the classifier during its use is a supervised algorithm. It is hence necessary to label run-time data. Two strategies can be used: an implicit labeling strategy that implicitly validates the recognized label in the absence of correction from the user; and an explicit labeling strategy, that solicits the user to obtain data true label.

The drawback of the implicit strategy is that, each time the user doesn't correct the recognized label, the user has to cancel/undo the wrong command that has been executed and then he can re-try to do the intended one. Furthermore, in case of non-corrected error, the classifier is trained with a wrong label, which deteriorates its model. To limit the execution of un-intended commands, and to be sure to learn with correct label, the system can solicit explicitly the user. However, it seems obvious that soliciting the user after each command would be very tedious for the user.

The strategy we use in this paper is to explicitly solicit the user to label the data samples that have a high probability to be mis-classified, and that it is very important to learn from. This kind of sample is difficult to recognize, as a consequence, it's very important to be able to learn from it. In these cases of confusion, the system solicits the user to obtain data true label. In other cases, the recognized label is implicitly validated if the user continues his task in the application. We also offer to users the possibility to explicitly correct the proposed label if he doesn't agree with the system proposal. We use the classifier confidence measure to select the data samples that aren't well described by the classifier model, and from which it will be very beneficial to learn. Considering these difficult samples, the classifier can learn from the gestures that are complex to recognize, and for which it would have probably made a mistake. As a consequence, the rejection strategy has a great influence on the classifier training process. The more data are rejected, the more data are available to train the classifier. Besides, learning from rejected data is very beneficial for the classifier model. Our objective is to improve our classifier recognition performances as much as possible, but without soliciting the user too often.

To detect conflicting classes, we use confusion reject principles [12]. Normally confusion reject is done using system output [13]. However, we try to detect confusion, to evaluate our model quality, at a very early stage of the learning process. As a result, inference rules conclusions are still unstable and are not suitable for confusion detection. Instead, we detect potential confusion on the membership to the different prototypes, which are much more stable at this early stage of the learning process. Even if every prototype participates in the recognition process of every class, each prototype has been created and is mainly associated with a single class. We use that fact to detect confusion when some gesture activates multiple prototypes at the same time.

To detect confusing classes, we define a confidence measure, and an associated confidence threshold. We flag a class as confusing when the confidence of its last gesture is below the threshold (when having learned on all the previous samples).

A. Confidence Measure

We use the Mahalanobis distance to compute the distance of a data sample $\mathbf{x}$ to prototype $C^{(i)}$ (defined by a center $\boldsymbol{\mu}^{(i)}$ and covariance matrix $\Sigma^{(i)}$).

$$distance(C^{(i)}, \mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top (\Sigma^{(i)})^{-1} (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top \quad (16)$$

From this distance, we compute a similarity measure that is smoother than prototype activation.

$$similarity(C^{(i)}, \mathbf{x}) = \frac{1}{1 + distance(C^{(i)}, \mathbf{x})} \quad (17)$$

With this similarity measure, we compute the similarity of a data sample to each prototype and take s_{first} and s_{second} as the first and the second highest similarity value. We compute system confidence as:

$$confidence = \frac{s_{first} - s_{second}}{s_{first}} \quad (18)$$

A data sample is then signaled as confusing when its confidence is below a certain threshold.

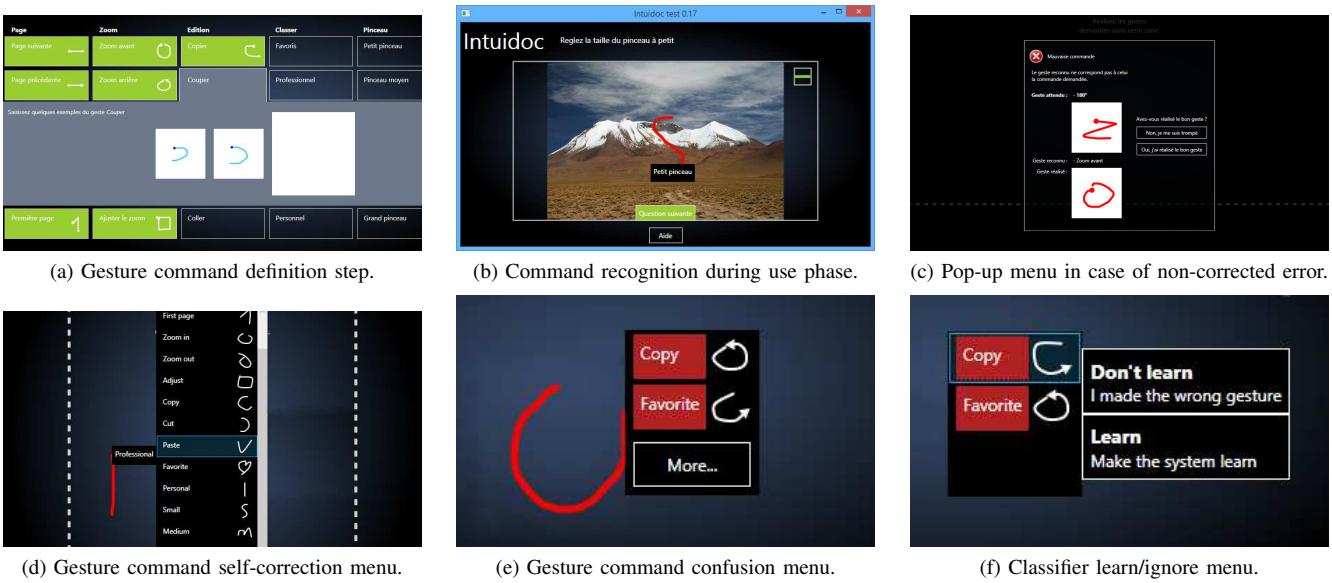
B. Threshold Selection

The optimization of the threshold below which we estimate gestures as confusing is multi-objective. One wants to maximize both classifier performance and accuracy:

$$Performance = N_{Correct}/N_{Total} \quad (19)$$

$$Accuracy = N_{Correct}/(N_{Correct} + N_{Errors}) \quad (20)$$

where $N_{Correct}$ is the number of correctly classified gestures, N_{Errors} is the number of incorrectly classified gestures, and N_{Total} is the total number of gestures. As the threshold increases, the number of rejected gestures raises and the number classification errors reduces.A high threshold will yield many rejections, which will increase system accuracy, whereas a low threshold will yield only a few rejections, which will increase system performance. There is a trade-off between the classifier performance and accuracy.



(a) Gesture command definition step.

(b) Command recognition during use phase.

(c) Pop-up menu in case of non-corrected error.

(d) Gesture command self-correction menu.

(e) Gesture command confusion menu.

(f) Classifier learn/ignore menu.

Fig. 2. Testing application screenshots.

To solve this trade-off, we must define the cost of an error of classification, and the cost of a rejection. During command definition step, a rejection will make the user change the command gesture (or draw more samples). During the utilization of gesture commands, an error of classification will force the user to correct the system. Our goal when detecting conflicting classes is to reduce classification errors during system use. However, we don't want to signal too many samples as conflicting, which would risk discouraging users from using our system.

The rejection threshold is quite difficult to estimate in our applicative context where we are learning from very few data samples. Moreover, each user chooses his own set of symbols/gestures, which may be quite complex to learn by the classifier. The rejection threshold must be automatically estimated for each user and his custom gesture set, and computed in a sufficiently simple and robust manner to yield good results with the few available data.

To estimate the rejection threshold, we chose to initialize our classifier on two of the three initialization samples per class, and to measure the recognition confidence on the third sample. We then compute the mean μ_{reco} and standard deviation σ_{reco} of the correctly recognized data and set the rejection threshold to one standard deviation below the mean: $threshold = \mu_{reco} - \sigma_{reco}$.

IV. Experimentation

This section is dedicated to gesture command experimentation, and to the evaluation of the impact of the strategy of using interactions based on the confusion reject capability of the classifier.

A. Experimental Protocol

We evaluated our system in a real context of gesture commands utilization, so that we obtain realistic results from users as well as from the recognizer. To do so, we designed a testing application: a picture viewer/editor with customizable gesture commands. This application has eighteen basic commands, grouped into six families, to manipulate pictures (like "next", "zoom in", "copy", etc.). We then designed a protocol that simulates a real use of this application to evaluate the impact of user interaction strategies. Our protocol was divided into four phases: an initialization phase to define gestures for each command (see Figure 2a), a first evaluation phase, a utilization phase and a second evaluation phase. During each phase (apart from the initialization phase), users were asked to do several commands one at the time.

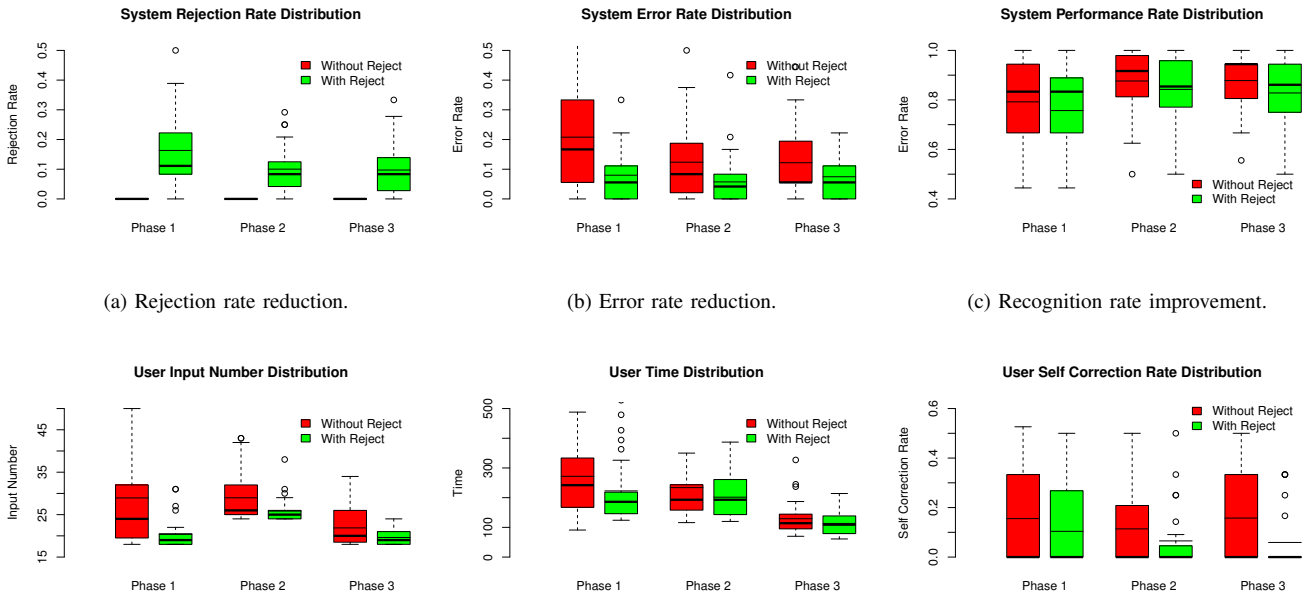
a) Phase 0: initialization: Users were asked to choose a gesture for each of the eighteen commands of the testing application (see Fig. 2a); and to repeat their gestures a few times to provide some initial training samples for the recognizer.

b) Phase 1: first evaluation phase: During this phase, users were asked to draw each of the eighteen commands once in random order.

c) Phase 2: utilization phase: This phase simulates a potential real use of our testing application. A total of twenty-four commands were asked in random order, some commands were asked thrice, some twice, some once and some were not asked. For this utilization phase, a help menu [14] was available to let users improve their memorization of their gesture commands.

d) Phase 3: second evaluation phase: During this phase, users were again asked to draw each of the eighteen commands once in random order.

Each time a gesture is drawn, the recognized command is executed and the command label is displayed at the end of the gesture stroke for three seconds (see Figure 3c). The label allows the user to access a small menu (see Figure 2d) to change the executed command if it isn't the one that is asked. When a gesture is rejected by the classifier, a small menu is displayed with the most probable labels (see Figure 2e), and the user has to select the correct one. Whenever the user corrects the classifier, either when self-correcting by clicking



(a) Rejection rate reduction.

(b) Error rate reduction.

(c) Recognition rate improvement.

(d) Number of inputs reduction.

(e) Time reduction.

(f) Self-correction reduction

Fig. 3. Error rate, recognition rate, number of inputs, time spent, self-correction rate and lies rate distribution for the two groups on the three phases.

on the recognized label or when a gesture his rejected, he is asked whether the classifier has to learn this correction or not (see Figure 2f). The classifier should learn when the mistake comes from the system, and forget the correction when the user has drawn a wrong gesture (corresponding to another command than the intended one). Whenever the user fails to notice and correct a wrong command, a pop-up (see Figure 2c) is being displayed at the end of the three seconds and ask if it is the user or the recognizer that made the mistake. The user has to re-do the command until it is correctly executed or corrected. All drawn gestures are used to incrementally train the recognizer, including wrong recognition that the user has failed to correct, to be as close as possible of a real utilization of a gesture command system.

A first group of users has done this experimentation without having explicit solicitations by the system based on confusion reject. User could only interact with the system by using spontaneous self-correction (by clicking on the recognized label). The learning process also used the implicit validation to learn from non-corrected samples. Users from this group were not alerted when the recognizer confidence was low. A second group of user has done this experimentation with the new user interaction strategy. The system automatically solicited users when confusion rejects were detected. Users could still do self-correction interaction when an error wasn't rejected and the system also learned by implicit validation.

We had 63 persons, from 20 to 57 year-old, participating in our test and everyone of them was used to manipulate computers in daily life. The first group (reference) contains 31 users, while the second group, with rejection based user interactions, contains 32 users.

TABLE I. REJECTION OVERALL PERFORMANCES.

	Recognized Gestures	Misclassified Gestures	Total
Non-Rejected Gestures	1750	42	1792
Rejected Gestures	86	162	248
Total	1836	204	2040

B. Rejection Performances

Rejection performances are presented in Table I. First, 65.3% of the rejected data would have been mis-classified if not rejected. The rest is rejected because the recognizer confidence is low, even if they are correctly recognized, and it is important to learn from those samples with the correct label. Moreover 79.4% of the classification errors are rejected, which is very good, height mis-classification over ten are avoided.

C. Recognition Performance Improvement

The error and rejection rates are presented Figure 3a and 3b. Rejecting data samples with low confidence (12.0% in average) allow to reduce the error rate from 15.1% to 7.06% (53% of relative reduction). We can notice that the rejection rate decreases with time as the classifier discrimination capacities improve, and that the error rate has already converged at the first test. Using rejection based user interactions allow to fasten the learning process of the classifier by improving the classifier learning supervision (progression from Phase 1 to Phase 3).

The performance rates are presented Figure 3c. The performance of the classifier is the rate of correctly labeled data, considering rejected data as non-labeled. There are no major differences between the two groups. The proportion of rejected data in the second group is compensated by the improvement of the classifier recognition performances.

D. User Interaction Improvement

By limiting the number of recognition errors, confusion driven user interactions allow to reduce the number of inputs the user has to draw to correctly execute the desired command (see Figure 3d). In particular, users of group 2 (with reject) draw 17.7% less gestures in average (65.8 instead of 79.8 for the whole experiment) than users of group 1, not using confusion based user interactions.

Consequently, time spent by users (see Figure 3e) is 15.3% lower when using confusion based user interactions (8.96 instead of 10.58 minutes). Even if soliciting the user every time a gesture is rejected by the classifier add several user interactions, it is much quicker for the user to validate/correct the classifier than to cancel a wrong command and do it again.

E. User Attention Deterioration

Figure 3f presents the rate of user corrections when the classifier makes a mistake. It appears clearly in this figure that the group 2 (with reject) suffers from the "user out-of-the-lop syndrome". Users of group 2 only correct 7.72% on average of the classifier remaining mistakes. This illustrates a drawback of our strategy based on an explicit solicitation of the user: this strategy tends to make users over-confident in the system so that they do less and less self-corrections.

This point is quite problematic because when a recognition error is not corrected, the classifier is trained with a wrong label which deteriorates its model. Hopefully, using confusion based user interactions reduces the number of recognition errors and limits the classifier model deterioration. For all that, we want to try to address this interesting problem in our future work to optimize the user and system cross-learning process.

V. Conclusion

We have presented a new method for increasing the efficiency of personalized gesture commands on pen-based devices. Such gesture commands require an evolving classifier that can learn from very few data samples and learn incrementally. We have explained how our system is able to explicitly solicit the user instead of making a mistake, which save the user from canceling/undoing an un-intended command before re-doing the intended one. Moreover, it allows to train on-line our recognizer *Evolve*– a first order fuzzy inference system – by explicitly supervising the labeling process with the user.

We have presented a complete and realistic experiment of gesture commands and we have studied the impact of our rejection based interaction method. This new Man-Machine interaction process give rise to a cross-learning situation where users have to learn and memorize command gestures and the classifier has to learn and recognize drawn gestures. In particular, we have shown that using explicit solicitation of the user (based on confusion reject) allows to significantly reduce the error rate and accelerate the on-line learning process. Moreover, rejection based user interactions improve gesture command efficiency by limiting the number of gestures drawn by users and reducing the time needed to complete the experiment.

However, rejection based interactions don't solve completely the "user out-of-the-loop problem". It actually tends to make users over-confident in the system and reduce their vigilance. Future work should address this point to maintain users vigilance and encourage them to spontaneously correct the system when necessary, which is essential to improve the classifier performance as much as possible.

Acknowledgment

The authors would like to thank Romain Lagneau for conducting all the user experiments during his training period within the IntuiDoc team at the IRISA Laboratory.

References

[1] J. O. Wobbrock, M. R. Morris, and A. D. Wilson, "User-defined gestures for surface computing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 1083–1092. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518866

[2] J. Yang, J. Xu, M. Li, D. Zhang, and C. Wang, "A real-time command system based on hand gesture recognition," in *2011 Seventh International Conference on Natural Computation (ICNC)*, vol. 3, 2011, pp. 1588–1592.

[3] P. Y. Li, N. Renau-Ferrer, E. Anquetil, and E. Jamet, "Semi-customizable gestural commands approach and its evaluation," in *2012 International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2012, pp. 473–478.

[4] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," in *Proceedings of the 20th annual ACM symposium on User interface software and technology*, ser. UIST '07. New York, NY, USA: ACM, 2007, pp. 159–168. [Online]. Available: http://doi.acm.org/10.1145/1294211.1294238

[5] A. Almaksour and E. Anquetil, "Improving premise structure in evolving takagi-sugeno neuro-fuzzy classifiers," *Evolving Systems*, vol. 2, no. 1, pp. 25–33, 2011. [Online]. Available: http://link.springer.com/article/10.1007/s12530-011-9027-0

[6] D. B. Kaber and M. R. Endsley, "Out-of-the-loop performance problems and the use of intermediate levels of automation for improved control system functioning and safety," *Process Safety Progress*, vol. 16, no. 3, pp. 126–131, 1997. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/prs.680160304/abstract

[7] D. A. Norman, "The 'problem' with automation: inappropriate feedback and interaction, not 'over-automation'," *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, vol. 327, no. 1241, pp. 585–593, 1990, PMID: 1970904.

[8] E. Lughofer, *Evolving fuzzy models: incremental learning, interpretability, and stability issues, applications*. VDM Verlag Dr. Mller, 2008.

[9] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *Systems, Man, and Cybernetics, IEEE Transactions on*, vol. 15, no. 1, pp. 116–132, 1985.

[10] P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1462–1475, 2008.

[11] P. Angelov and D. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 484 – 498, 2004.

[12] C. Chow, "On optimum recognition error and reject tradeoff," *IEEE Transactions on Information Theory*, vol. 16, no. 1, pp. 41 – 46, 1970.

[13] H. Ishibuchi and T. Nakshima, "Fuzzy classification with reject options by fuzzy if-then rules," in , *The 1998 IEEE International Conference on Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence*, vol. 2, 1998, pp. 1452 –1457 vol.2.

[14] P. Li, A. Delaye, and E. Anquetil, "Evaluation of continuous marking menus for learning cursive pen-based commands," 2011, pp. 217–220. [Online]. Available: http://hal.inria.fr/hal-00741295